Curvature- and Model-based Hatching of Patient-specific Muscle Surfaces

Christian Tietjen, Rocco Gasteiger, Alexandra Baer, Bernhard Preim

Department of Simulation and Graphics, Otto-von-Guericke-University Magdeburg
`tietjen@isg.cs.uni-magdeburg.de`

Abstract. In this paper, we introduce a new method for hatching patient-specific, anatomic 3D surface models, in particular hatching of muscle surfaces. However, the method is transferable to other structures, *e. g.*, , vessels. Since 3D surface models are extracted from binary segmentation masks, the curvature orientation is rough and noisy and the final hatching direction is falsified. The goal of this work is the correction and the completely automated computation of the hatching direction.

1 Introduction

In medical textbooks, didactically prepared illustrations are used to comprehensively present complex scenarios. Specific visualization goals are a better object discrimination and perception of the object's shape [1]. In this illustrations, muscle fibers are often depicted by using hatches.

There are different approaches to map hatching on surfaces [2, 3, 4]. All approaches depend on a well-defined curvature field for the hatching line alignment. A texture-based approach is most suitable, because it is frame-coherent (the hatching lines are attached on the surface), real-time capable and it is possible to zoom in very closely. This is not achieved by image- or object-based approaches.

Hatching is also helpful to improve interactive 3D visualizations. To uncover hidden objects, primarily transparency is applied. Though, it is challenging to find an adequate degree of transparency. The hiding object's shape is either not recognizable or the hidden objects are barely visible. Therefore, opacity mapping may help when the texture is aligned to the muscle fibers [5].

This work focusses on hatching of muscular structures, namely the automatic generation of correctly aligned hatching lines on surfaces derived from segmented clinical CT/MRI datasets. So far, previous work only relied on curvature information [6]. Since 3D surface models are extracted from binary segmentation masks, the curvature orientation is rough and noisy and the final hatching direction is falsified, even though the surface is smoothed. This fact may also cause singularities (*e. g.*, vortices), which also falsify the final direction.

Furthermore, the mathematically correct curvature does not correspond to the muscle fiber direction in most cases. Additional model-based knowledge is needed to improve the curvature field. Therefore, the muscle fiber direction has to be ascertained. The final direction reflects the shape of the surface and the direction of the muscle fibers.

2 Materials and Methods

Our approach to map hatching textures on muscle surfaces consists of three steps: curvature approximation, determination of the muscle fiber direction and combination with the curvature approximation, as well as surface parameterization to map hatching textures.

Step 1: At first we smooth an instance of the anatomical surface by Laplacian smoothing. On this instance, we estimate the two principle curvature directions used for the original vertices (Fig. 1(a)). We employ quadratic surface fitting at each vertex v for curvature approximation [7]. This method needs only three incident vertices (inclusive v) in contrast to five or nine neighbors for more accurate approximation operators. In practice we found that the results provided by this method are accurate enough [8]. Additional, we apply an optimization method presented in [2] to smooth principle curvature directions at unstable vertex locations.

Step 2: We implemented two methods to approximate the direction of muscle fibers. The first one is performed by a skeletonization in voxel space according to Selle et al. [9]. We extract a 3D skeleton from previous skeletonization which consists of linked skeleton voxels s_i. For each v, we determine the nearest s_i and calculate a vector $\hat{r}_v$ which is perpendicular to the local skeleton direction of s_i. The skeletonization method is only appropriate for almost cylindrical muscles to determine a meaningful centerline. Hence, we implemented a main axis computation and select the longest axes to approximate the direction of fiber (Fig. 1(b)).

After the previous step, two principle curvature directions and a model-based preferential direction $\hat{r}_v$ at each vertex v are available. In the following, we merge these information to a single vector field with one vector for each face. For the initial vector field, we select at each vertex v the principle direction which has the smallest angle to $\hat{r}_v$ and denote this direction as $\hat{f}_v$. We improve the alignment of each $\hat{f}_v$ to the preferential direction $\hat{r}_v$ and to the incident neighborhood. The optimization is processed by rotating $\hat{f}_v$ in direction of $\hat{r}_v$. For each optimization step, we rotate $\hat{f}_v$ into the direction of $\hat{r}_v$ and calculate the average angular deviation to all $\hat{f}_i$ in the incident neighborhood. Rotation stops if the angle between $\hat{f}_v$ and $\hat{r}_v$ as well as the angular deviation becomes larger in comparison to the previous step. This corresponds to a local minimum. Finally, we assign to each face the average vector of its vertices and the resulting vector field is used in the following surface parameterization (Fig. 1(c)).

Step 3: The mapping of 2D textures on 3D surfaces requires a surface parameterization which has a minimum of distortions. Due to the rough and complex surface of the medical volume datasets, common two-pass methods are not appropriate and cause strong texture distortions. We employ *lapped patches* from Praun et al. [10] and extend their approach to a global parameterization. In contrast to Praun et al., we create each patch depending on its neighbored patch and map it relative to its neighbor in a global parameter space. This results in a complete unfolded surface. Finally, we treat patch boundaries to avoid almost all texture seams. In contrast to [10] no texture blending for minimizing visible texture seams is necessary at these vertices.

The entire patch growing is based on the surface geometry and the underlying vector field from step 2. For the first patch, we select an arbitrary face as reference face D_{ref} and define a local basis consisting of the face vector $\hat{f}$, face normal $\hat{n}$ and vector $\hat{s}$ which is perpendicular to both vectors. We employ a change of bases into the xy-plane of the world coordinate system to align $\hat{x}\|\hat{s}$, $\hat{y}\|\hat{f}$ and $\hat{z}\|\hat{n}$. This plane is in the following our global parameter space. Further faces around D_{ref} are rotated in the tangent plane of D_{ref} and transformed into the parameter space according to the local base of D_{ref}. The transformation is performed relative to the reference face to preserve the topology in the parameter space. The patch growing is continued until a projection window W around D_{ref} is covered or a certain stop criteria are satisfied. The projection window is associated to the complete texture. The stop criteria consider the avoiding of strong texture distortions and variations in the direction of the vector field around D_{ref}. Finally, a global outline of all parameterized faces is calculated.

Based on the start patch, further patches are created to cover the surface. For each new patch, an inner face from the global outline list is selected and denoted as new reference face D_{ref} for the current patch. Analog to the start patch, we create a new basis and perform a transformation to the parameter space. In contrast to the start patch, we translate the new D_{ref} to the position in its previous patch to perform a continued unfolding at the current patch boundaries. The further patch growing is similar to the start patch but the rotated patch faces are transformed relative to the new position of D_{ref}. The surface parameterization is finished, if every face belongs to at least one patch which results in a complete unfolded mesh.

Some faces are part of more than one patch and each patch has a slight varying alignment to its neighbored patches. Hence, several vertices in overlapped regions have multiple parameter coordinates. Initially, this results in visible texture seams at patch boundaries. We resolve this seams by averaging correspondent parameter coordinates of affected vertices in the parameter space. This simple approach achieves an continued texture over patch boundaries without strong visible texture distortions. The graphic hardware scales the resulting parameter coordinates between the interval $[0, 1]$ and converts them into texture coordinates for texture access.

3 Results

The method described in this paper makes it possible to render real-time capable and frame-coherent hatching textures on anatomical surfaces (Fig. 1(d)). Since texture rendering is directly supported by the graphics hardware, there are no noticeable runtime-differences in comparison to rendering without texture.

User interaction is reduced to the adjustment of a few parameters, for which default values could by found. For surface smoothing, we select $w = 0.7$ and one iteration for the Laplacian smoothing. The weighting factor w specifies the strength of vertex displacement and is defined between 0.0 and 1.0. The optimization method of [2] has two thresholds

$$\frac{\max(|\kappa_1|, |\kappa_2|)}{\min(|\kappa_1|, |\kappa_2|)} > \delta_{\text{ratio}} \land \max(|\kappa_1|, |\kappa_2|) > \delta_{\text{reliable}} \tag{1}$$

to describe parabolic vertices where curvature estimations are assumed as stable. Principle curvature directions of non-parabolic (i. e. unstable) vertices are aligned at these stable directions. We determined the average of each parameter as default value. The values κ_1 and κ_2 in Eq. 1 defines the principle curvatures values. The last parameter affects the maximum allowed angle deviation of the direction vector of a reference triangle D_{ref} to a new triangle candidate for adding it to the given patch. We denote this parameter as $\alpha_{\text{divergence}}$ and select 5° as appropriate value which has a good trade off between texture minimization and memory consumption.

Finally, the parameterization of the surface has to be performed only one time per anatomical structure in a pre-processing step and the time needed for pre-processing is also maintainable.

4 Discussion

The goal of the paper at hand was to develop a fully automatic and real-time capable approach to hatch surfaces derived from clinical 3D-datasets in interactive visualizations. For that purpose, we combined a curvature- and model-based vector field and mapped the final hatching on the surface by using texture. The generated vector field is robust against rough and noisy anatomical surfaces. Only a

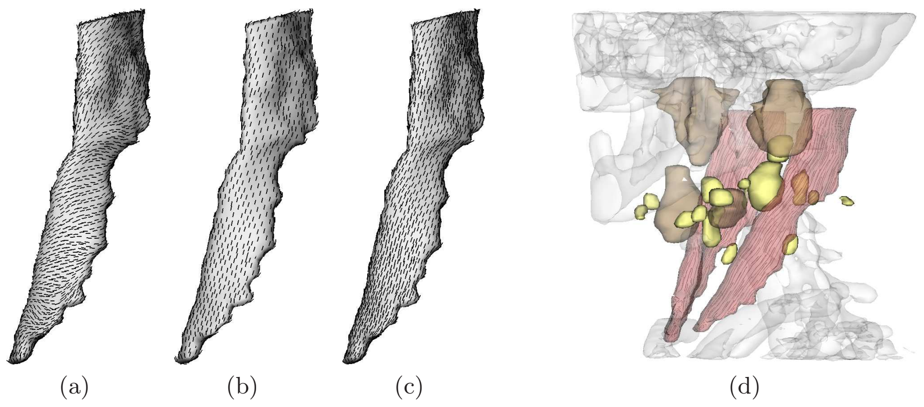

Fig. 1. (a) Visualization of the curvature field. (b) Visualization of the preferential direction. (c) Combination of both fields. (d) Applied texture mapping. Due to the *Opacity Mapping*, the muscle and the lymph nodes lying behind are distinguishable

few singularities and discontinuities were observed. Regarding the parameterization, these critical points caused visible texture distortions. Especially triangles with differing size lead to a strong averaging of the parameter coordinates. These problem may be solved by regularizing the size of the triangles.

For bended muscles, the longest main axes does not approximate the initial fiber direction sufficiently. The creation of geometric centerline by adaptively dividing the main axes and moving the new junctions in the local geometric center of the model should achieve satisfying results.

Furthermore, completely different approaches for defining preferential direction might be helpful for hatching other types of surfaces. It is possible to extract the inner skeleton of structures, *e. g.*, , the middle line of vessels [9], to use it as model-based vector field. The definition by an intermediate surface is also possible.

We also aim at evaluating the usefulness of the generated hatched surfaces for medical visualizations, according to Interrante et al. [1]. We are particularly interested in the impact on the perception of the object's shape and the differentiation of separate structures.

Acknowledgement. We thank MeVis Research for providing advanced MeVisLab features. Christian Schulz we thank for the analysis and implementation of the curvature approximation. This work was carried out in the framework of a project supported by the Deutsche Forschungsgemeinschaft (DFG) (Priority Programme 1124, PR 660/3-1).

References

1. Interrante V, Fuchs H, Pizer SM. Conveying the 3D shape of smoothly curving transparent surfaces via Texture. IEEE Trans Vis Comput Graph. 1997;3(2):98–117.
2. Hertzmann A, Zorin D. Illustrating smooth surfaces. In: SIGGRAPH; 2000. p. 517–26.
3. Zander J, Isenberg T, Schlechtweg S, et al. High quality hatching. Comput Graph Forum. 2004;23(3):421–40.
4. Praun E, Hoppe H, Webb M, et al. Real-time hatching. In: SIGGRAPH; 2001. p. 579–84.
5. Krüger A, Tietjen C, Hintze J, et al. Interactive visualization for neck dissection planning. In: EuroVis; 2005. p. 295–302.
6. Kälberer F, Nieser M, Polthier K. QuadCover: Surface parameterization using branched coverings. Comput Graph Forum. 2007;26(3):375–84.
7. Goldfeather J. Understanding Errors in Approximating Principal Direction Vectors. University of Minnesota, Department of Computer Science and Engineering; 2003.
8. Schulz C. Approximation von Krümmungsinformation für die medizinische Visualisierung. University of Magdeburg: Dept. of Computer Science; 2005.
9. Selle D, Preim B, Schenk A, et al. Analysis of vasculature for liver surgery planning. IEEE Trans Med Imaging. 2002;21(11):1344–57.
10. Praun E, Finkelstein A, Hoppe H. Lapped textures. In: SIGGRAPH; 2000. p. 465–70.